The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Subgraph-preconditioned Conjugate Gradients for Large Scale SLAM

Frank Dellaert, Justin Carlson, Viorela Ila, Kai Ni and Charles E. Thorpe

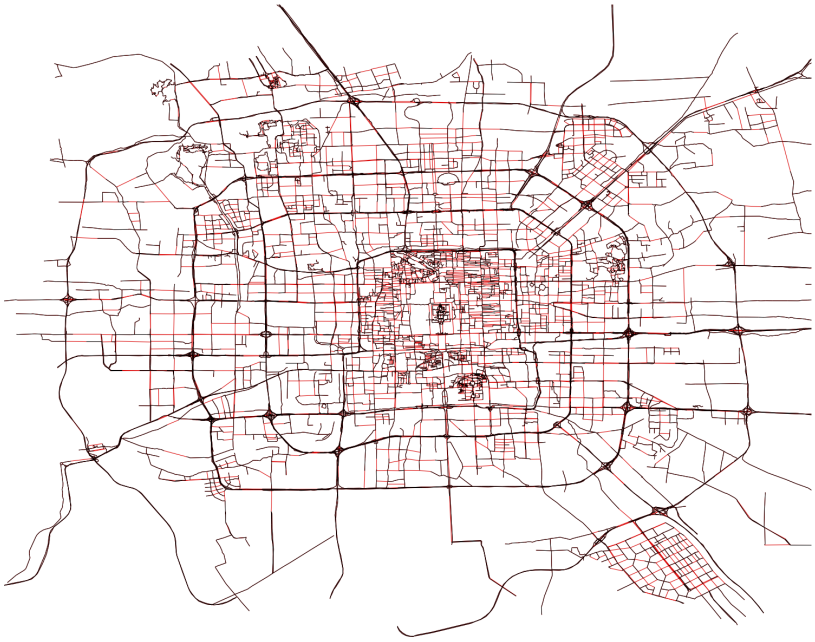


Fig. 1: *The main idea in this paper is to combine the advantages of direct and iterative methods: we identify a subgraph that can easily be solved using direct methods, and use that as a preconditioner in conjugate gradients. This is illustrated above with a map of Beijing, where the subgraph is a spanning tree (in black), and the remaining loop-closing constraints are shown in red.*

***Abstract*— In this paper we propose an efficient preconditioned conjugate gradients (PCG) approach to solving large-scale SLAM problems. While direct methods, popular in the literature, exhibit quadratic convergence and can be quite efficient for sparse problems, they typically require a lot of storage and efficient elimination orderings to be found. In contrast, iterative optimization methods only require access to the gradient and have a small memory footprint, but can suffer from poor convergence. Our new method, *subgraph preconditioning*, is obtained by re-interpreting the method of conjugate gradients in terms of the graphical model representation of the SLAM problem. The main idea is to combine the advantages of direct and iterative methods, by identifying a sub-problem that can be easily solved using direct methods, and solving for the remaining part using PCG. The easy sub-problems correspond to a spanning tree, a planar subgraph, or any other substructure that can be efficiently solved. As such, our approach provides new insights into the performance of state of the art iterative SLAM methods based on re-parameterized stochastic gradient descent. The efficiency of our new algorithm is illustrated on large datasets, both simulated and real.**

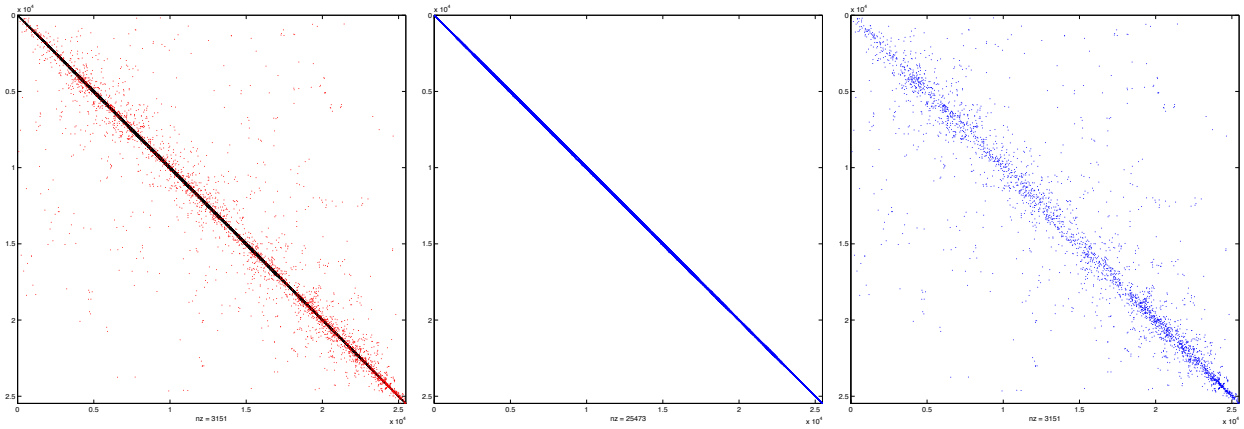
Frank Dellaert, Viorela Ila, and Kai Ni are affiliated with the Georgia Institute of Technology, Atlanta (dellaert,vila, nikai)@gatech.edu, and gratefully acknowledge the support from the National Science Foundation, Awards 0713162 and 0448111 (CAREER). Viorela Ila is also at the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, and has been partially supported by the Spanish Ministry of Science and Innovation under the *Programa Nacional de Movilidad de Recursos Humanos de Investigación*. Justin Carlson and Charles E. Thorpe (justinca, cet)@cs.cmu.edu are affiliated with Carnegie Mellon Qatar, Doha, Qatar

I. Introduction

The state of the art in simultaneous localization and mapping (SLAM) can be categorized into filtering-based solutions, sub-mapping, sparse direct methods, and iterative methods. Early solutions to the SLAM problem relied on the extended Kalman filter (EKF) to estimate the position of landmarks and robot pose and their associated covariances. However, due to their quadratic memory and computational cost, EKF-based solutions are limited to small areas [1].

Filtering itself has been shown to be inconsistent when applied to the inherently non-linear SLAM problem [2], i.e., even the average solution taken over a large number of experiments diverges from the true solution. Since this is mainly due to the fact that linearization points are far from the true solution in a filtering framework, there has recently been considerable interest in applying nonlinear optimization to the smoothing SLAM problem [3], [4], [5], which iteratively solves a sequence of linear systems using the previous estimates as linearization points. Smoothing approaches also have another important advantage, in that the linear systems to be solved in each iteration remain sparse over time [5], which enable SLAM solutions that have better scaling properties for larger problems.

In non-linear optimization, the key computation is solving a linear system, done with either a direct or iterative method.



(a) T+C (b) T (c) C

Fig. 2: Block adjacency matrices for Beijing simulation: (a) the sparsity pattern of the original information matrix corresponding to Figure 1, with the non-zero blocks for the spanning tree in black and the remaining constraints in red; (b) the spanning tree T alone, with 25473 constraints; (c) the remaining 3151 loop closing constraints C.

While **direct methods** exhibit quadratic convergence and can be quite efficient for sparse problems, they typically require a lot of storage as well as efficient elimination orderings to be found. For very large mapping problems, especially with many loop closures, the computational cost and memory requirements can still make these methods impractical. A common approach to reduce the computational cost is to split the large scale SLAM problems into smaller subproblems. In hierarchical mapping [6], [7], [8], [9], [10] the map elements are grouped respect to their spatial distribution by build local submaps that are integrated into a global map. Local maps are limited to a bounded number of poses/landmarks and thus, their construction can be carried out in constant time. Consequently, hierarchical mapping shifts the complexity to the map joining phase.

In contrast, **iterative methods** only require access to the gradient and have a small memory footprint, but can suffer from poor convergence. Popular iterative methods in are Gauss-Seidel relaxation methods on the constraint graph. These were pioneered by [11] and [12], and later also adapted in [13], [14] to relax a sparsified information matrix. Several strategies were proposed to cope with slow convergence rates, by adapting several methods from the iterative optimization literature. [15] introduced the use of preconditioned conjugate gradient (PCG) descent, using incomplete Cholesky factors as preconditioners. In [16], Frese et. al. present a Multi-Level Relaxation (MLR) algorithm, inspired by multigrid methods in [17]. However, the current state of the art in pose-graph relaxation stems from the work by [3] whose main contribution is a parameterization of the global poses in terms of increments along the trajectory. Grisetti [4], [18] adopted this incremental parameterization as well, but instead defines it on a spanning tree. Both lines of work use stochastic gradient descent (SGD) to minimize the resulting objective functions in terms of incremental poses.

Subgraph Preconditioned Conjugate Gradients

The main idea is to combine the advantages of direct and iterative methods, by identifying a sub-problem that can be easily solved using direct methods, and solving for the remaining part using PCG. In particular, we propose a new method inspired by looking at preconditioned gradient descent in novel way, in the context of graphical models.

Our new method, *subgraph preconditioned conjugate gradients* (SPCG) is obtained by re-interpreting conjugate gradients in terms of the graphical model representation of SLAM. The idea is to break the problem down in two parts: a part that easy to be solved like for example an "odometry prior", and a second part which distributes the loop closing constraint. The main idea is that the latter is more efficiently solved using a preconditioned method. When considering graphSLAM, the preconditioner can be a spanning tree, or more generally we can allow loops and find a *planar* or "fat" subgraph, or any other substructure that can be efficiently solved. As such, our approach provides new insights into the performance of state of the art iterative SLAM method based on re-parameterized stochastic gradient descent.

The idea is illustrated with an example shown in 1. We simulated a pose-constraint graph for the city of Beijing, using publicly available data, to generate a graph with $25,474$ nodes and $28,624$ edges. Below we discuss how we split up a large problem with measurement matrix A into two systems, A_1 and A_2. In the Beijing case we chose $A_1 = T$, a spanning tree shown in the figure in black, whereas the remaining "loop closing constraints" A_2 are shown in red. The corresponding $25,474 \times 25,474$ adjacency matrices are show in Figure 2. They are shown with columns ordered by depth in the spanning tree, with the root at the far right. This is the optimal ordering for solving the $25,473$ constraints in T in $O(n)$, whereas only $3,151$ of the original $28,624$ constraints (11%) will be iterated over using PCG.

II. Background

We follow the statement of the SLAM problem in [5], obtaining the nonlinear least-squares objective function

$$f(\theta) = \frac{1}{2}\sum_i |h_i(\theta) - z_i|^2$$

We linearize around a linearization point θ_0 to get a new, linear least-squares problem in x with objective function

$$f(x) = \frac{1}{2}|Ax - b|^2 \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$ is the measurement Jacobian and x an n-dimensional vector in the tangent space $T(\theta_0)$ around θ_0. After solving, the original unknowns θ can be obtained from the solution $\hat{x}$ using the exponential map:

$$\theta = e^{\hat{x}}\theta_0 \tag{2}$$

In the case where the unknowns θ live in a vector space $\mathbb{R}^n$, the exponential map reverts to simple vector addition $\theta = \theta_0 + \hat{x}$. However, this is typically *not* the case in SLAM, as rotations live in non-linear manifolds $SO(2)$ or $SO(3)$.

In a **direct method**, we use either QR or Cholesky factorization to solve for the least squares solution $\hat{x}$. In particular, using QR factorization we can factor the Jacobian A, augmented with b as an extra column, to obtain

$$\begin{bmatrix} A & b \end{bmatrix} = Q \begin{bmatrix} R & c \\ 0 & e \end{bmatrix} \tag{3}$$

with R an $n\times n$ upper triangular matrix, Q an $m\times m$ rotation matrix, and c and e vectors. We can then rewrite the error

$$\frac{1}{2}|Ax - b|^2 = \frac{1}{2}|Rx - c|^2 + \frac{1}{2}e^T e$$

where $\frac{1}{2}e^T e$ is the least-squares residual error, and the solution $\hat{x} = R^{-1}c$ is found efficiently using back-substitution.

Another strategy is to form the system of normal equations

$$\Lambda x = \eta \tag{4}$$

where $\Lambda \triangleq A^T A$ is the *information matrix* and $\eta \triangleq A^T b$ is the *information vector*. We then apply Cholesky decomposition, a specialized version of Gaussian elimination for symmetric positive definite matrices, to factor the information matrix as $\Lambda = R^T R$. The *square root information matrix* R is the same upper-triangular matrix as in (3). We then solve for $c = R^{-T}\eta$, after which we obtain $\hat{x}$ as before by back-substitution. Cholesky decomposition is more efficient if $m \ggg n$, but is numerically less well-conditioned.

In both cases the posterior probability density $P(x|z)$ on x given the measurements z can be approximated by a Gaussian around $\hat{x}$ with covariance matrix $\Sigma = \Lambda^{-1}$, as

$$\begin{aligned} P(x|z) &\approx k\exp\left\{-\frac{1}{2}|Rx - c|^2\right\} \\ &= k\exp\left\{-\frac{1}{2}(x-\hat{x})^T\Lambda(x-\hat{x})\right\} \end{aligned}$$

which in turn induces a probability density $P(\theta|z)$ via (2).

Algorithm 1 Method of Conjugate Gradients (CG)

Start with $x_0 = 0$, $g_0 = -A^T b$, $d_0 = -g_0$, and iterate:

$$\alpha_k = -d_k^T g_k / \|Ad_k\|^2$$
$$x_{k+1} = x_k + \alpha_k d_k$$
$$g_{k+1} = g_k + \alpha_k A^T(Ad_k)$$
$$\beta_{k+1} = \|g_{k+1}\|^2 / \|g_k\|^2$$
$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$$

An alternative to direct methods is to employ an **iterative method** to solve the system of normal equations (4). An iterative method starts from an estimate x_0 and finds the best estimate along a *search direction* d,

$$x = x_0 + \alpha d$$

where α is a scalar step-size. Substituting into (1) we obtain

$$f_\alpha(\alpha) = \frac{1}{2}|A(x_0 + \alpha d) - b|^2$$

Setting the directional gradient of f_α with respect to α to zero, we obtain an expression for the optimal step-size α:

$$\alpha = -d^T g(x_0) / \|Ad\|^2$$

where the *gradient* $g(x)$ of $f(x)$ is defined as

$$g(x) \triangleq f'(x) = A^T(Ax - b) = \Lambda x - \eta$$

When we iterate this and, at each iteration k, move in the direction of the negative gradient, i.e., $d_k = -g_k$, we obtain the *steepest descent* algorithm. However, this is sub-optimal, as the steepest descent directions have contributions that repeat earlier search directions, leading to slow convergence. The solution is to make each search direction Λ-conjugate with respect to the previous one [19], or

$$d_k^T \Lambda d_{k+1} = 0$$

This can be done by taking the gradient direction and subtracting out some of the previous search direction d_k:

$$d_{k+1} = -(g_{k+1} - \beta_{k+1} d_k) \text{ with } \beta_{k+1} = \|g_{k+1}\|^2 / \|g_k\|^2$$

As we start with $x_0 = 0$ we have $g_0 = -\eta = -A^T b$, and it is easy to show that $g_{k+1} = g_k + \alpha_k \Lambda d_k$. Note that for efficiency we first compute Ad_k and then $\Lambda d_k = A^T(Ad_k)$. The resulting *conjugate gradient* (CG) method is listed as Algorithm 1 and is equivalent to the least-squares CG variant (CGLS) described by Björck in [20, page 288].

III. Approach

A. Preconditioned CG as Reparameterizing

Preconditioning is a technique where instead of solving $\Lambda x = \eta$ we solve $M_1^{-1}\Lambda M_2^{-1} M_2 x = M_1^{-1}\eta$, with M_1 and M_2 respectively the $n \times n$ *left and right preconditioning matrices*. With the right choice of preconditioning matrices

the new system $M_1^{-1}\Lambda M_2^{-1}$ can exhibit dramatically faster convergence properties.

In this paper we consider the original least-squares formulation (1), and in this context the *preconditioned conjugate gradient* (PCG) method can be explained as reparameterizing the unknowns x in terms of new unknowns y [20]. If we set $y = Px$ in (1) we obtain a new objective function in y,

$$\bar{f}(y) = \frac{1}{2}\left|AP^{-1}y - b\right|^2$$

with corresponding normal equations $P^{-T}\Lambda P^{-1}y = P^{-T}\eta$, a preconditioned system with $M_1 = P^T$ and $M_2 = P$.

We will consider a less customary "non-central" re-parameterization $y = Px - q$, or $x = P^{-1}(q + y)$, leading to

$$\bar{f}(y) = \frac{1}{2}\left|AP^{-1}y - \left(b - AP^{-1}q\right)\right|^2$$

We use the same PCG method expect we now iterate on y, use a different starting gradient $\bar{g}_0$, and Λ is replaced by $P^{-T}\Lambda P^{-1}$. This can be most easily implemented by calling Algorithm 1 with $\bar{A} = AP^{-1}$ and $\bar{b} = b - AP^{-1}q$. We then recover the original solution as $\hat{x} = P^{-1}(q + \hat{y})$.

B. Subgraph Conditioning

The main idea in the paper is to combine the advantages of direct and iterative methods in SLAM, by identifying a sub-problem that can be easily solved using direct methods, and solving for the remaining part using PCG. For the sub-problem we consider a subset of the measurement constraints in A that form a spanning tree, planar subgraph, or any subgraph that allows an efficient elimination order.

In particular, suppose that by cutting measurements rows from A the new problem A_1 becomes easy to solve using a direct method. We can then rewrite the criterion (1) as

$$f(x) = \frac{1}{2}\left|A_1x - b_1\right|^2 + \frac{1}{2}\left|A_2x - b_2\right|^2 \qquad (5)$$

where A_1 corresponds to the easy part and A_2 to the measurements that are cut. Let us then consider the solution $\bar{x} \triangleq R_1^{-1}c_1$ to A_1, with corresponding Gaussian log likelihood $\frac{1}{2}\left|R_1x - c_1\right|^2$. Hence, we can rewrite the criterion (5) as

$$f(x) = \frac{1}{2}\left|R_1x - c_1\right|^2 + \frac{1}{2}\left|A_2x - b_2\right|^2 \qquad (6)$$

where the first term acts as a prior on x derived from A_1, and the second term represents the error from the A_2 part. Now let us re-parameterize (precondition!) x in terms of the whitened deviation y from the prior:

$$y = R_1x - c_1 = R_1(x - \bar{x})$$

By substituting $x = \bar{x} + R_1^{-1}y$ in (6), we obtain

$$\bar{f}(y) = \frac{1}{2}\left|y\right|^2 + \frac{1}{2}\left|A_2R_1^{-1}y - \bar{b}_2\right|^2$$

with $\bar{b}_2 \triangleq b_2 - A_2\bar{x}$. The interpretation is intuitive: the first term penalizes deviation y from the subgraph solution $\hat{x}$ and implicitly encodes the measurement error in A_1, whereas the second term represents the measurement error in A_2. If $y = 0$, that error is simply $\bar{b}_2$, the prediction error for measurements in A_2 using the approximate solution $\bar{x}$.

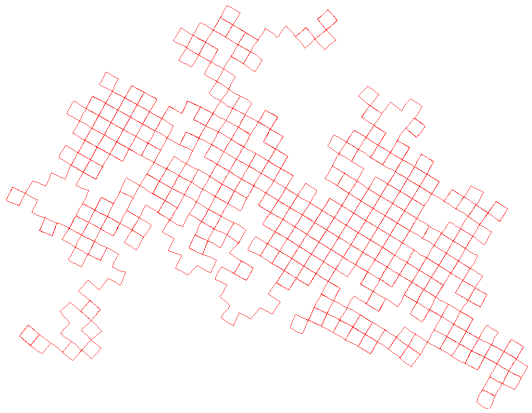


Fig. 3: Manhattan world optimized graph with10000 vertices and 64312 edges.

The implementation of the idea is quite easy: we simply call the original LSCG conjugate gradients (Algorithm 1) on the following system (which requires solving R_1):

$$\bar{A} = \begin{bmatrix} I \\ A_2R_1^{-1} \end{bmatrix} \text{ and } \bar{b} = \begin{bmatrix} 0 \\ \bar{b}_2 \end{bmatrix}$$

After convergence, we recover the solution by

$$\hat{x} = \bar{x} + R_1^{-1}\hat{y}$$

IV. Experiments and Results

A. Simulation Experiments

Subgraph preconditioned conjugate gradient (SPCG) combines the advantage of both direct and iterative methods. The subgraph used as preconditioner plays an important role in the convergence of the system. Here we analyze different types of subgraphs by applying the proposed subgraph preconditioning method to simulated and read data. The results are obtained with a Matlab implementation running under Mac OS X on an Intel Core 2 at 2.4 GHz.

We applied our method to a simulated dataset, courtesy of Grisetti [4]. This dataset simulates a robot navigating in a Manhattan-like environment, looping around square blocks several times and establishing many links (see Figure 3). The entire dataset contains 10000 vertices connected by 64312 edges. Figure 4 (left) shows the time comparison between solving the system using direct methods (red), using the implementation of the Conjugate Gradient method from the Algorithm 1 (green) and our SPCG method (red+green). The latter is a combination of a direct part (solving for $\bar{x}$, shown in red) and the iterative LSCG (iterating on y, shown in green). For each of the four SPCG variants, we also show the ratio $f \triangleq \frac{|A_1|}{|A_2|}$ of the size of the "easy" problem (matrixA_1, solved with the direct method) and the size of the "hard" problem (matrix A_2, solved with CG). If this ratio is infinite, $f = Inf$, the size of the $A_2 = 0$, so A_1 consists of the entire graph and is solved using direct methods. If this ratio is zero $f = 0$, the whole system is solved using CG. For this

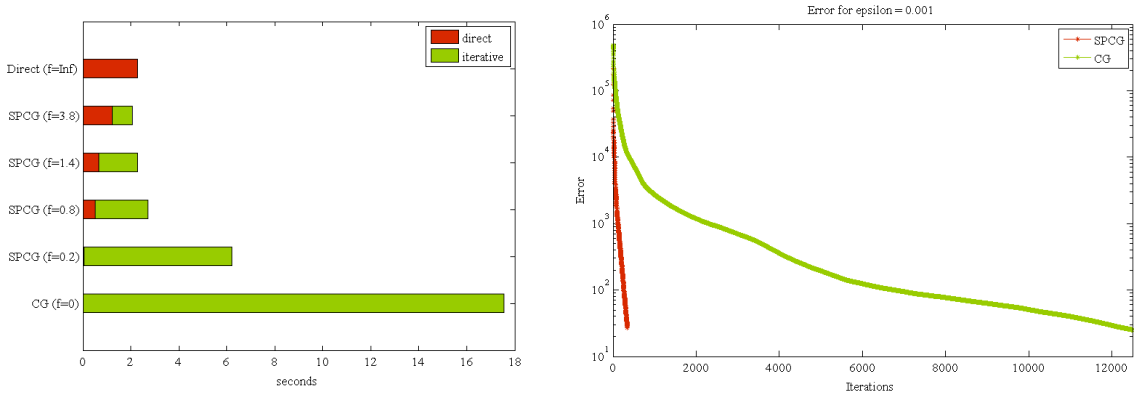


Fig. 4: Manhattan world results. (left) Time comparison averaged over 10 runs for solving the graph using direct methods (red), CG (green) and our new SPCG method (red+green). Different subgraph sizes are compared: $f = 1.4$, $f = 0.8$ and $f = 0.2$, where f is the ratio between the of the "easy" and "hard" part. (right) Comparison of the convergence rate corresponding to the CG and SPCG's iterative part.

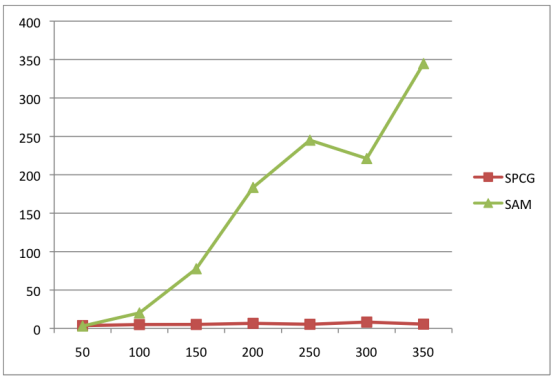


Fig. 5: Time comparison between SAM and SPCG. The X-axis shows the number of landmarks visible from 1000 poses, and the Y axis shows timing results for both SAM and SPCG.

dataset, when we used the spanning tree subgraph as A_1, this ratio was equal to $f = 0.2$. The other values for the ratio f correspond to adding supplementary edges to the spanning tree to decrease the size of the "hard" part.

For this comparison we used a direct method solver as implemented in the SuiteSparse toolbox. Notice that the SuiteSparse, currently the most efficient implementation for direct solvers for sparse problems, is multi-threaded and takes full advantage of the dual core architecture. Nevertheless, our single-threaded and non-optimized Matlab implementation of the SPCG solver proved to be 10% faster for large graphs such as the ones in Figure 3.

To assess the performance of the SPCG when applied to a very challenging scenario we simulated a robot taking 1000 steps in the 2D plane. The pose was modeled using the usual $SE(2)$ parametrization, and the robot executed a random walk on a regular lattice, moving forward 1 m, turning 90 degrees left, or turning 90 degrees right. For each step a simulated odometry measurement was created. In addition to these $N = 1000$ odometry constraints we also generated M landmarks, also in $SE(2)$, visible from anywhere in the world. From any position, one landmark is chosen at random and a measurement constraint is generated, hence the total graph contains $(M + N - 1)$ edges and $(M + N)$ vertices.

We compared SPCG with Smoothing and Mapping (SAM) [5]. The simulation described above is a particularly bad scenario for direct methods such as SAM, as the graph will be densely connected and large cliques will be generated for any ordering. In [21] it was explained that certain graphs, such as planar graphs, can be optimized in $O(n^{1.5})$, but if landmarks are seen from anywhere (a situation we call the Eiffel-tower scenario) this can completely destroy the sparsity of the graph. However, it is hypothesized that because the number of constraints is the same, an iterative method should exhibit constant timing.

Figure 5 shows time comparison between SAM and SPCG. The x-axis shows the number M of landmarks visible from N =1000 poses, and the Y axis shows timing results for both SAM and SPCG. As one can see, SAM performs quite well until $N = 100$, when the generated cliques will cause the inversion of 300×300 matrices. This becomes worse, although it depends on the specific graph that was generated (which explains the dip for $M = 350$). In contrast, the timing results for SPCG stay flat over the entire range.

B. Experiments with Real Data

We also evaluated and tested the proposed technique on real data, using the data set collected at the Intel Research Lab (Seattle), available at Radish dataset repository. This dataset includes odometry and laser scans readings. The laser scans are used to generate sensor-based odometry and to assert loop closures, by aligning them using an ICP scan matching algorithm. The resulting graph of constraints has 729 vertices and 3071 edges.

In real SLAM applications, the most natural way to form a subgraph is to consider the odometry chain which can easily be constructed while the robot navigates and builds the map of the environment. Figure 6 shows in red the odometry tree

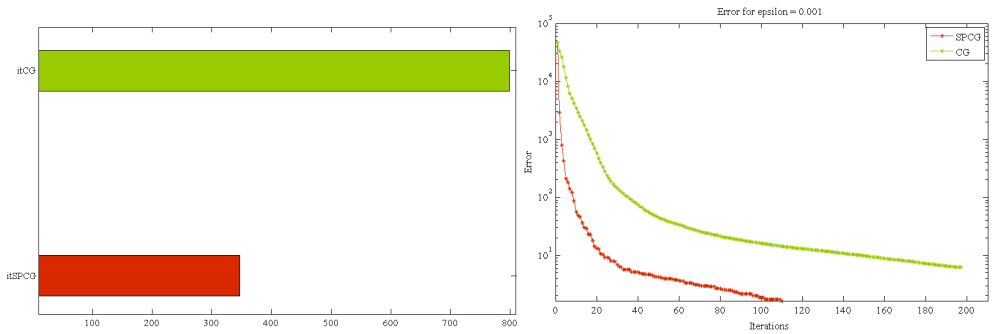


Fig. 7: Intel data set results. (left) Comparison with the CG in number of iterations. (right) Comparison of the convergence rate corresponding for the CG and SPCG's iterative part.

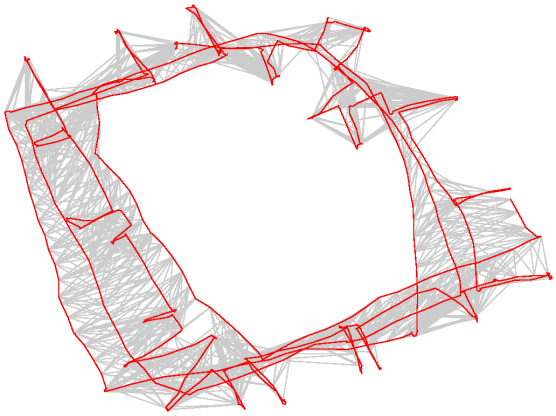


Fig. 6: The (unoptimized!) Intel data set, showing the subgraph corresponding to the odometry backbone.

over the entire graph plotted in gray. In Figure 7 (left and right) we compared the number of iterations and the error drop during the iterative solving of the preconditioned system using odometry chain and the CG.

V. Conclusions

We propose an efficient preconditioned conjugate gradients (PCG) approach to solving large-scale SLAM problems. Our new method, *subgraph preconditioning*, is obtained by re-interpreting the method of conjugate gradients in terms of the graphical model representation of the SLAM problem, and combines the advantages of direct and iterative methods. In future work, we hope to demonstrate that more advanced subgraphs such as thin junction trees or planar graphs can improve performance even more.

References

[1] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Intl. J. of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1987.

[2] S. Julier and J. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 4, 2001, pp. 4238–4243.

[3] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2006.

[4] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Robotics: Science and Systems (RSS)*, Jun 2007.

[5] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec 2006.

[6] J. Tardós, J. Neira, P. Newman, and J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *Intl. J. of Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[7] C. Estrada, J. Neira, and J. Tardós, "Hierarchical SLAM: Real-time accurate mapping of large environments," *IEEE Trans. Robotics*, vol. 21, no. 4, pp. 588–596, Aug 2005.

[8] K. Ni, D. Steedly, and F. Dellaert, "Tectonic SAM: Exact; out-of-core; submap-based SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Rome; Italy, April 2007. [Online]. Available: http://www.cc.gatech.edu/ dellaert/pubs/Ni07icra.pdf

[9] L. M. Paz, P. Pinies, J. D. Tardós, and J. Neira, "Large scale 6DOF SLAM with stereo-in-hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[10] S. Huang, Z. Wang, and G. Dissanayake, "Exact state and covariance sub-matrix recovery for submap based sparse eif slam algorithm," May 2008, pp. 1868–1873.

[11] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, San Francisco, CA, 2000.

[12] A. Howard, M. Matarić, and G. Sukhatme, "Relaxation on a mesh: a formalism for generalized localization," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Wailea, Hawaii, Oct 2001, pp. 1055 – 1060.

[13] S. Thrun and Y. Liu, "Multi-robot SLAM with sparse extended information filters," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*. Sienna, Italy: Springer, 2003.

[14] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *Intl. J. of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[15] K. Konolige, "Large-scale map-making," in *Proc.* 21^{th} *AAAI National Conference on AI*, San Jose, CA, 2004.

[16] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Trans. Robotics*, vol. 21, no. 2, pp. 196–207, April 2005.

[17] W. Briggs, *A Multigrid Tutorial*. SIAM, 1987.

[18] G. Grisetti, C. Stachniss, and W. Burgard, "Non-linear constraint network optimization for efficient map learning," *Trans. on Intelligent Transportation systems*, 2009.

[19] M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, December 1952.

[20] A. Björck, *Numerical methods for least squares problems*. SIAM, 1996.

[21] P. Krauthausen, F. Dellaert, and A. Kipp, "Exploiting locality by nested dissection for square root smoothing and mapping," in *Robotics: Science and Systems (RSS)*, 2006.